REMARKS

Claims 1, 5, 8 and 10 have been amended. Minor grammatical corrections have been made to the specification. Reexamination and reconsideration are respectfully requested.

In the Office Action, Claims 1-10 were rejected as being anticipated by Sweeney et al (U.S. Patent No. 6,230,314B1). Applicants respectfully traverse this rejection in view of the clarifying amendments made to the independent claims and the following remarks.

In independent Claims 1, 5, 8 and 10, Applicants have clarified the nature of their "object-oriented functions" to make clear the stark differences between the presently claimed invention and the Sweeney reference . As recited in Claim 1, for example, Applicants have made clear that the function removing means checks the specification information derived from the specification analysis means and, based on a predetermined function removal rule, removes an unnecessary function from a set of object-oriented functions by which members are realized. In other words, program information is generated by excluding unnecessary member realization methods from among the object-oriented member realization methods (see page 3, lines 14-20 and page 18, lines 2-11).

The object-oriented functions in the present invention include, for example, a function of dynamic generation of an instance (Claim 3), a function of a virtual function (Claim 2) and a function of an inheritance. These object-oriented functions by which members are realized are discussed in Applicants' specification in connection with Figures 5, 9, 10, 11, 13, 14 and 17.

Importantly, Applicants' claimed object-oriented functions do not imply the members themselves, which are generic terms such as variables and routines contained in classes in an object-oriented program and are well known by definition in the C++ language, and also correspond, for example, to the "Boolean" and "void" shown in Figure 3 of Applicants' specification.

Rather, as would be readily understood by those skilled in the art from Applicants' specification, the defined object-oriented functions are the mechanisms or methods of realizing the members (see page 18, lines 2-11).

In accordance with Applicants' novel invention, through the use of the object-oriented function removing means, when program code is generated from the object-oriented specification information, no program code is generated for an unneeded member realization method, *i.e.*, an object-oriented function. This eliminates wasted memory space and allows for a reduction in the required memory capacity. As a result, an inexpensive embedded control system can be achieved along with a high speed program code generating method.

By contrast, Sweeney discloses a method for <u>program</u> transformation in which <u>unnecessary components or members themselves</u> are removed from objects of a program (Column 1, lines 43-46 and Column 2, lines 8-19). This operation is completely different from Applicants' invention, which removes an object-oriented function by which the member itself is realized. In other words, whereas in Sweeney the unneeded members themselves are removed, in the present invention the unneeded member realization methods, including the dynamic generation of instances, virtual functions and inheritances, are removed.

This distinction is extremely important. For example, in Sweeney, all of the realization methods for the members are coded regardless of whether they are necessary or not. Hence, even though some may be unnecessary, the codes are nevertheless stored in a memory in any event (uselessly). This thus requires a large memory capacity that cannot be generally accommodated in an inexpensive embedded control system of the type according to Applicants' invention. As discussed above, Applicants' invention overcomes this problem by excluding the unnecessary member realization method itself from the generated program code.

In view of the foregoing, Applicants submit independent Claims 1, 5, 8 and 10 are patentable over Sweeney. In that regard, independent Claims 5 and 8 have been amended to clarify that the object-oriented function is a function by which a member is realized. Method Claim 10 has been likewise amended.

Because Sweeney does not teach or suggest selecting or removing member realization methods (object-oriented functions) it is respectfully submitted that the independent claims are patentable thereover. Further, Claims 2-4, 6-7 and 9 depend respectively from the independent claims and are also submitted to be patentable.

Regarding the comments on Applicants' Information Disclosure Statement filed February 22, 1999, Applicants request reconsideration as the relevance of JP 6-266562 was disclosed in Applicants' background of the invention section on pages 1 and 2. 37 C.F.R. §1.98(a)(3)(i) and (ii) specifically states that the concise explanation may be incorporated in Applicants' specification. An English-language translation of the document need only be filed if it is within the possession, custody, or control of the §1.56(c) designee.

Regarding the Japanese patents referenced by the Examiner in the Office Action, Applicants note that these are not all assigned to the present assignee. Nonetheless, Applicants submit an additional Information Disclosure Statement enclosing English-language abstracts of those documents. Also, a non-prior art paper by some of the present inventors is submitted as part of the IDS.

For the foregoing reasons, Applicants submit Claims 1-10 are patentable over the prior art of record. An early notice to that effect is solicited. If there are any questions regarding this amendment or the application in general, a telephone call to the undersigned would be appreciated since this should expedite the prosecution of the application for all concerned.

If necessary to effect a timely response, this paper should be considered as a petition for an Extension of Time sufficient to effect a timely response, and please charge any deficiency in fees or credit any overpayments to Deposit Account No. 05-1323 (Docket #381NP/47598).

Respectfully submitted,

[illegible] 32,169

pro Gary R. Edwards
Registration No. 31,824

CROWELL & MORING, LLP
Intellectual Property Group
P.O. Box 14300
Washington, DC 20044-4300
Telephone No.: (202) 624-2500
Facsimile No.: (202) 628-8844
JDS/GRE:kms

VERSION WITH MARKINGS TO SHOW CHANGES MADE TO THE SPECIFICATION

Please amend the first and second full paragraphs on page 1 as follows:

The present invention relates to an automatic software generating system, and in particular, to a software generating system for automatically generating [a] software from an object-oriented specification which features an improved code efficiency.

A code generating system generates [a] software from an object-oriented software statement. Generally, a specification described by an operator is entered via an input device, then the specification is analyzed, and a program code is generated and output.

Please amend the third full paragraph on page 14 as follows:

In case a table cannot be read from the memory device 104 in step 702, [s] a standard generation pattern information is output in step 704. In case the table could have been read from the memory device 104 in step 702, it is determined in step 705 whether to use or not to use any particular function according to the function select item having been read. When it is determined to use, a standard generation pattern information defined by the function select item is output in step 707. When judged not to use, a code generation pattern information defined by the function select item is output in step 706.

VERSION WITH MARKINGS TO SHOW CHANGES MADE TO THE CLAIMS

Please amend Claims 1, 5, 8 and 10 as follows:

1. (Amended) A software generation system comprising:

a specification analysis means which analyzes an object-oriented specification for deriving specification information;

a function removing means which checks said specification information derived by said specification analysis means by collating with a function removal rule which is predetermined, and removes a function which becomes unnecessary from a set of object-oriented functions <u>by which members are realized</u>, for generating <u>from the specification information</u> program information excluding the unnecessary function; and

a code generation means for generating a code according to said program information obtained by said function removing means.

5. (Amended) A software generation system comprising:

an input means for inputting a specification described as diagrammatic information, and selecting an object-oriented function <u>by which members are realized</u> to utilize;

an analysis means for analyzing said specification entered via said input means;

a function selection means which outputs pattern information for use in generating a code on the basis of a result of analysis by said analysis means and according to said object-oriented function selected; and

a code generation means for generating a program code of said specification analyzed according to the pattern information output from said function selection means.

8. (Amended) A software generation system comprising:

a specification analysis means which analyzes an object-oriented specification for deriving specification information;

an analysis result display means for displaying a status of use of an object-oriented function <u>by which a member is realized</u> from said specification information;

an input means whereby to select an object-oriented function to utilize;

a function memory means for storing a function selected via said input means;

a program information generation means for generating program information on the basis of said specification information derived by said specification analysis means and using said function selected and stored in said memory means; and

a code generation means for generating a code on the basis of said program information obtained by said program generation means.

10. (Amended) A software generation method comprising the steps of:

analyzing an object-oriented specification entered;

generating program information using object-oriented functions <u>by which members are realized</u> without unnecessary functions according to a predetermined function removing rule; and

generating a code of said specification analyzed on the basis of said program information.